

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-074253

(43)Date of publication of application : 15.03.2002

(51)Int.Cl. G06F 19/00
G06F 3/00
G06F 17/60

(21)Application number : 2000-267676

(71)Applicant : TOSHIBA CORP
TOSHIBA SYSTEMS
DEVELOPMENT CO LTD

(22)Date of filing : 04.09.2000

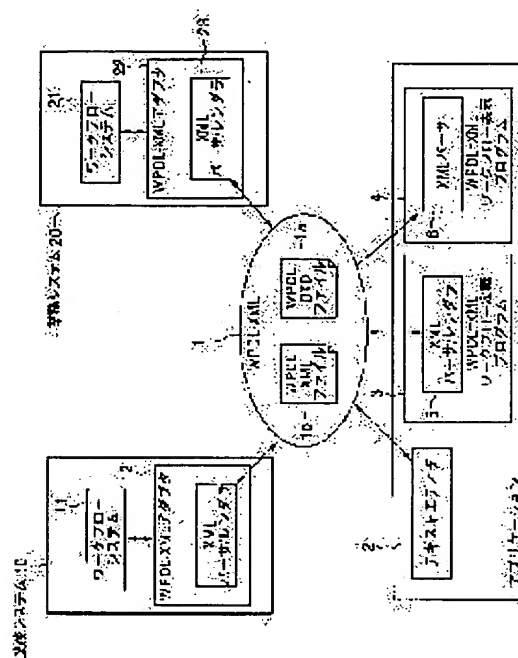
(72)Inventor : TERASHITA YOSUKE
YOSHIMOTO TAKEHIRO

(54) METHODS FOR DEFINING, DISPLAYING, REUSING AND CONVERTING WORK FLOW

(57)Abstract:

PROBLEM TO BE SOLVED: To make easily performable cooperation between different kinds of work flow systems.

SOLUTION: This work flow defining method is characterized by providing a step to display icon parts for an operation concerning work flow definition on a screen, a step to define the work flow by arranging the icon parts on the screen, a step to generate a text character string for describing work flow definition in XML on the basis of document type definition which is stipulated for describing work flow definition in XML and a step to output the generated text character string.



LEGAL STATUS

[Date of request for examination]

21.02.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号
特開2002-74253
(P2002-74253A)

(43)公開日 平成14年3月15日(2002.3.15)

(51)Int.Cl. ⁷	識別記号	F I	データベース(参考)
G 0 6 F 19/00	3 0 0	G 0 6 F 19/00	3 0 0 N 5 B 0 4 9
3/00	6 5 1	3/00	6 5 1 A 5 E 5 0 1
17/60	1 6 2	17/60	1 6 2 C

審査請求 未請求 請求項の数6 OL (全 20 頁)

(21)出願番号	特願2000-267676(P2000-267676)	(71)出願人	000003078 株式会社東芝 東京都港区芝浦一丁目1番1号
(22)出願日	平成12年9月4日(2000.9.4)	(71)出願人	000221100 東芝システム開発株式会社 東京都港区芝浦1丁目1番1号 東芝ビルディング
		(72)発明者	寺下 陽介 東京都港区芝浦1丁目1番1号 株式会社 東芝本社事務所内
		(74)代理人	100077849 弁理士 須山 佐一

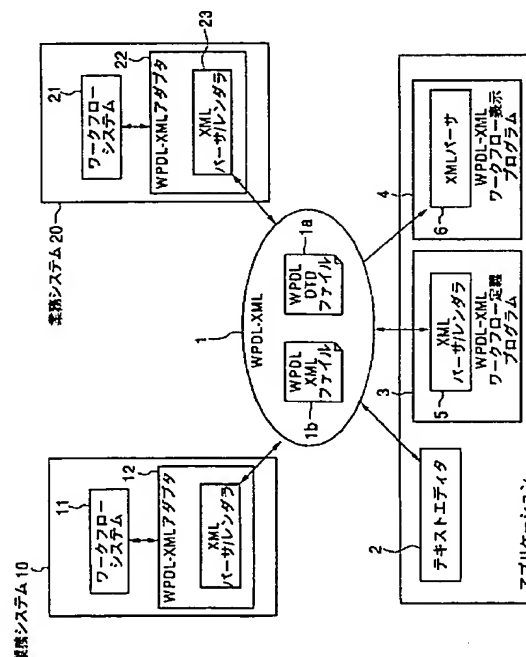
最終頁に続く

(54) 【発明の名称】 ワークフロー定義方法、ワークフロー表示方法、ワークフロー再利用方法、ワークフロー変換方法

(57)【要約】

【課題】 異種ワークフローシステム間の連携を容易に行えるようにする。

【解決手段】 このワークフロー定義方法は、ワークフロー定義に関する操作作用のアイコン部品を画面に表示する段階と、アイコン部品が画面上に配置されることでワークフローを定義する段階と、ワークフローの定義をXMLで表記するために規定した文書型定義に基づき、ワークフローの定義をXMLで表記したテキスト文字列を生成する段階と、生成されたテキスト文字列を出力する段階とを有することを特徴とする。



【特許請求の範囲】

【請求項1】 ワークフロー定義に関する操作のアイコン部品を画面に表示する段階と、
前記アイコン部品が前記画面上に配置されることでワークフローを定義する段階と、
ワークフローの定義をExtensible Markup Languageで表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Languageで表記したテキスト文字列として出力する段階とを有することを特徴とするワークフロー定義方法。

【請求項2】 ワークフロー処理が実行されているシステムより、ワークフローの進捗状況の情報をExtensible Markup Languageで表記したテキスト文字列で取得する段階と、

取得されたテキスト文字列を、ワークフローの定義をExtensible Markup Languageで表記するために規定した文書型定義に基づき、ワークフローの形態で表示する段階とを有することを特徴とするワークフロー表示方法。

【請求項3】 あるワークフローシステムのワークフローをExtensible Markup Languageで表記したテキスト文字列を解析する段階と、

解析されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成する段階と、

生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成する段階と、

生成されたワークフローモデルを該当ワークフローシステムに登録する段階とを有することを特徴とするワークフロー定義変換方法。

【請求項4】 あるワークフローシステムのワークフローをExtensible Markup Languageで表記したテキスト文字列に変換する段階と、

変換されたテキスト文字列を他のワークフローシステムへ伝送する段階と、

伝送されてきたテキスト文字列を、自身のワークフローに変換する段階とを有することを特徴とするワークフロー再利用方法。

【請求項5】 ワークフロー定義に関する操作のアイコン部品を画面に表示する段階と、

前記アイコン部品が前記画面上に配置されることでワークフローを定義する段階と、

ワークフローの定義をExtensible Markup Languageで表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Languageで表記したテキスト文字列に変換する段階と、

変換されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成する段階と、

生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成する段階と、

生成されたワークフローモデルを該当ワークフローシステムに登録する段階とを具備したことを特徴とするワークフロー変換方法。

【請求項6】 クライアントコンピュータとサーバコンピュータとをネットワークを介して接続してなるクライアント・サーバシステムにおけるワークフローの変換方法において、

前記クライアントコンピュータにおいて、あるワークフローシステムのワークフローをExtensible Markup Languageで表記したテキスト文字列に変換する段階と、

変換されたテキスト文字列をネットワークを通じて前記サーバコンピュータへ伝送する段階とを有し、

前記サーバコンピュータにおいて、前記ネットワークからテキスト文字列が受信された場合、変換対称ワークフローシステム用のワークフロー変換手段を起動する段階と、

起動されたワークフロー変換手段が、受信された文字列を、それぞれのワークフローシステム用のワークフローモデルに変換する段階と、

変換されたワークフローモデルを、それぞれのワークフローシステムに登録する段階とを有することを特徴とするワークフロー変換方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、例えば複数の異なるワークフローシステムを連携させるためのワークフロー定義方法、ワークフロー表示方法、ワークフロー再利用方法、ワークフロー変換方法に関する。

【0002】

【従来の技術】従来から、ワークフロー管理システムとして、例えばInconcert（アメリカのTIBCO社商品名）等が良く知られている。

【0003】このようなワークフロー管理システムは、人から人への仕事の受け渡しのルールをコンピュータ上に定義し、仕事の受け渡しを自動化するためのソフトウェアである。

【0004】近年、上記以外にも様々なワークフロー製品が市場に存在しているが、ワークフロー定義（仕事の受け渡しのルール定義）の方法は、個々のワークフロー製品によって異なる。

【0005】また、インターネットなどのネットワークを利用した業務処理が急速に普及している現在では、ワークフローの利用範囲も拡大される傾向にある。

【0006】このようにワークフローの利用範囲が大きくなるに連れて、異なるワークフロー製品を統合する汎用的なワークフロー定義言語の必要が生じてきた。

【0007】そこで、ワークフロー標準化団体WFMC(Workflow Management Coalition)が発足し、汎用的なワークフロー定義言語として、Workflow Process Definition Language: WPDLが定義された。

【0008】

【発明が解決しようとする課題】しかしながら、WPDLが定義されたものの、このWPDLの言語処理プログラムを開発する負担などから、WPDLの実装を済ませているワークフロー製品は多くないのが現状である。

【0009】したがって、WPDLが実装されたワークフロー製品と既存のワークフロー製品とのシステム間連携は今後とも必要であり、このような連携技術を構築することが急務になっている。

【0010】本発明はこのような課題を解決するためになされたもので、異なるワークフローシステム間で容易にワークフローを連携させることのできるワークフロー定義方法、ワークフロー表示方法、ワークフロー再利用方法、ワークフロー変換方法を提供することを目的としている。

【0011】

【課題を解決するための手段】上記目的を達成するために、請求項1記載の発明のワークフロー定義方法は、ワークフロー定義に関する操作のアイコン部品を画面に表示する段階と、前記アイコン部品が前記画面上に配置されることでワークフローを定義する段階と、ワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Language で表記したテキスト文字列として出力する段階とを有することを特徴としている。

【0012】請求項2記載の発明のワークフロー表示方法は、ワークフロー処理が実行されているシステムより、ワークフローの進捗状況の情報をExtensible Markup Language で表記したテキスト文字列で取得する段階と、取得されたテキスト文字列を、ワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの形態で表示する段階とを有することを特徴としている。

【0013】請求項3記載の発明のワークフロー変換方法は、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列を解析する段階と、解析されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成する段階と、生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成する段階と、生成されたワークフローモデルを該当ワークフローシステムに登録する段階とを有することを特徴としている。

【0014】請求項4記載の発明のワークフロー再利用

方法は、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換する段階と、変換されたテキスト文字列を他のワークフローシステムへ伝送する段階と、伝送されてきたテキスト文字列を、自身のワークフローに変換する段階とを有することを特徴としている。

【0015】請求項5記載の発明のワークフロー変換方法は、ワークフロー定義に関する操作のアイコン部品を画面に表示する段階と、前記アイコン部品が前記画面上に配置されることでワークフローを定義する段階と、ワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Language で表記したテキスト文字列に変換する段階と、変換されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成する段階と、生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成する段階と、生成されたワークフローモデルを該当ワークフローシステムに登録する段階とを有することを特徴としている。

【0016】請求項6記載の発明のワークフロー変換方法は、クライアントコンピュータとサーバコンピュータとをネットワークを介して接続してなるクライアント・サーバシステムにおけるワークフロー変換方法において、前記クライアントコンピュータにおいて、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換する段階と、変換されたテキスト文字列をネットワークを通じて前記サーバコンピュータへ伝送する段階とを有し、前記サーバコンピュータにおいて、前記ネットワークからテキスト文字列が受信された場合、変換対称ワークフローシステム用のワークフロー変換手段を起動する段階と、起動されたワークフロー変換手段が、受信された文字列を、それぞれのワークフローシステム用のワークフローモデルに変換する段階と、変換されたワークフローモデルを、それぞれのワークフローシステムに登録する段階とを有することを特徴としている。

【0017】請求項1記載の発明では、画面上に表示されたアイコン部品が画面上に配置されることでワークフローを定義し、このワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Language で表記したテキスト文字列として出力するので、テキストエディタ等を使わずとも、ワークフローの定義をExtensible Markup Language で表記できる。

【0018】すなわち、画面上でのアイコン部品の配置操作、つまりドラッグアンドドロップの操作等でワークフローを定義でき、しかもこれと同時に、ワークフロー

定義のXML変換が自動的に行われるので、ユーザは、あるワークフローモデルについての知識さえあれば、そのワークフローモデルの表記法もXMLの表記法も知らなくても、XMLで表記したワークフローモデルを作成することができる。

【0019】請求項2記載の発明では、ワークフロー処理が実行されているシステムより、ワークフローの進捗状況の情報をExtensible Markup Language で表記したテキスト文字列で取得し、取得されたテキスト文字列を、予め規定された文書型定義に基づき、ワークフローの形態で表示するので、ユーザは、現在のワークフローの進捗状況をビジュアル的に確認することができる。

【0020】請求項3記載の発明では、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列を解析し、解析されたテキスト文字列を基に、所定言語のプログラムで編集可能のようにデータオブジェクトクラスのインスタンスを生成し、この生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成し、生成されたワークフローモデルを該当ワークフローシステムに登録するので、異種ワークフロー間のワークフロー連携が可能になる。

【0021】請求項4記載の発明では、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換し、この変換されたテキスト文字列を他のワークフローシステムへ伝送し、他のワークフローシステムでは、伝送されてきたテキスト文字列を、自身のワークフローに変換するので、一度、作成したワークフローモデルを他のワークフローシステムで再利用できるようになる。

【0022】請求項5記載の発明では、画面上に表示されたアイコン部品が画面上に配置されることでワークフローを定義し、このワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Language で表記したテキスト文字列に変換するので、テキストエディタ等を使わずとも、ワークフローの定義をExtensible Markup Language で表記できる。

【0023】また、変換されたテキスト文字列を解析し、解析されたテキスト文字列を基に、所定言語のプログラムで編集可能のようにデータオブジェクトクラスのインスタンスを生成し、この生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成し、生成されたワークフローモデルを該当ワークフローシステムに登録するので、異種ワークフロー間のワークフロー連携が可能になる。

【0024】請求項6記載の発明では、クライアントコンピュータにおいて、あるワークフローシステムのワー

クフローをExtensible Markup Language で表記したテキスト文字列に変換し、この変換されたテキスト文字列をネットワークを通じてサーバコンピュータへ伝送し、サーバコンピュータにおいてネットワークからテキスト文字列が受信された場合、変換対称ワークフローシステム用のワークフロー変換手段を起動し、起動されたワークフロー変換手段が、受信された文字列を、それぞれのワークフローシステム用のワークフローモデルに変換し、変換したワークフローモデルをそれぞれのワークフローシステムに登録するので、異なるワークフローシステム間のワークフロー同期をとることができる。

【0025】この結果、異なるワークフローシステム間で容易にワークフローを連携させることができる。

【0026】

【発明の実施の形態】以下、本発明の実施の形態を図面を参照して詳細に説明する。図1は本発明に係る一つの実施の形態のワークフロー連携論理モデル、図2は図1の論理モデルを具現化したワークフロー連携システムの一例を示す構成図である。

【0027】図1において、符号1は異なるワークフロー間でやりとりされるWPDL-XMLである。このWPDL-XML1は、WPDL-XMLの文書構造や意味を表現するために使用するタグ情報が記述されたWPDL-DTDファイル1aと、このWPDL-DTDファイル1aに従って記述されたWPDL-XMLファイル1bとからなる。DTDファイルは、個々のXMLファイルがどのようなタグや属性を持つべきかという規定として定義するファイルであり、上記WPDL-DTDファイル1aは、それをWPDLの定義に適合させたものである。XMLとは、Extensible Markup Languageの略である。DTDとは、Document TypeDefinitionの略である。WPDL-XMLファイル1bはデータによってその内容が異なるが、WPDL-DTDファイルの内容は固定である。

【0028】符号2～6はアプリケーションソフトウェアである。2はテキストエディタ(Windows付属のメモ帳など)である。テキストエディタ2は、テキストファイルであるWPDL-XMLファイル1bをWPDL-XMLの表記法に従って編集するためのツールである。3はWPDL-XMLワークフロー定義プログラムであり、ユーザがWPDL-XMLの表記法を知らなくてもGUI操作によりワークフローモデルの定義(ワークフローの流れの設定)を行え、この際、操作内容に応じてWPDL-XMLファイル1bを生成するプログラムである。4はWPDL-XMLワークフロー表示プログラムであり、読み込んだWPDL-XMLファイルを基にワークフロー情報をグラフィカルに表示するプログラムである。5はXMLパーサ/レンダラであり、XMLファイルの読み込み及び書き出しを行うプログラム(ソフトウェア)である。このプログラム(ソフトウェア)については市販のXMLパーサ/レンダラ製品を使用できる。6はXMLパーサであり、XMLファイルの読み込み、所定の形態、例えばワークフロー情報などに交換するプログラム(ソフトウ

ウェア)である。このプログラム(ソフトウェア)も市販のXMLパーサ製品を使用できる。

【0029】符号10は業務システムであり、独立してワークフローを実行するワークフローシステム11とWPDL-XMLアダプタ12とを備えたシステムであり、ワークフローシステム11を中心とし、この業務システム10特有の業務データベース(以下業務DBと称す)と業務アプリケーションとの連携を行う。WPDL-XMLアダプタ12は、WPDL-XMLファイル1bを読み込み、ワークフローシステム11への定義として登録し、また、ワークフローシステム11のワークフロー進捗情報を、WPDL-XMLファイル1bに書き出す、このシステム対応用のアダプタである。WPDL-XMLアダプタ12には、上記XMLパーサ/レンダラ5と同様のXMLパーサ/レンダラ13が備えられている。

【0030】符号20は業務システムであり、独立してワークフローを実行するワークフローシステム21とWPDL-XMLアダプタ22とを備えたシステムであり、ワークフローシステム21を中心とし、この業務システム20特有の業務DBと業務アプリケーションとの連携を行う。WPDL-XMLアダプタ22は、WPDL-XMLファイル1bを読み込み、ワークフローシステム21への定義として登録し、また、ワークフローシステム21のワークフロー進捗情報を、WPDL-XMLファイル1bに書き出す、このシステム対応用のアダプタである。WPDL-XMLアダプタ22には、上記XMLパーサ/レンダラ5と同様のXMLパーサ/レンダラ23が備えられている。

【0031】この図1の論理モデルを、クライアント・サーバシステムとして具体化した例が図2である。

【0032】図2に示すように、この例のクライアント・サーバシステムは、インターネットに接続されたクライアントコンピュータ31(以下クライアントPC31と称す)及びサーバ用コンピュータ32と、このサーバ用コンピュータ32とLAN等のイントラネット介して接続された個々にワークフロー処理を実行するワークフローサーバ用のコンピュータ34、37とからなり、クライアントPC31上のインターネットブラウザ42からサーバ用コンピュータ32にアクセスし、WPDL-XMLファイル1bを送受することで、各ワークフロー処理用のコンピュータ34、37とワークフローの連携を行うよう構成されている。

【0033】コンピュータ34には、ワークフローサーバ35とワークフロー定義データを格納するデータベース36とが設けられている。コンピュータ37には、ワークフローサーバ38とワークフロー定義データを格納するデータベース39とが設けられている。

【0034】クライアントPC31は、CPU、ROM、RAM等のメモリ、ハードディスクドライブ装置等の補助記憶装置、モニタ、キーボード等の入出力装置を有するコンピュータであり、例えばWindowsやMacintosh

h等のオペレーティングシステム(以下OSと称す)の環境下で動作するものである。ハードディスクドライブ装置には、WPDL-XML作成プログラム41、インターネットブラウザ42等のソフトウェアがインストールされている。WPDL-XML作成プログラム41は、ワークフロー定義情報ファイルであるWPDL-XMLファイル43を作成する。このWPDL-XMLファイル43は、WPDL-XML定義プログラム41の他、一般のテキストエディタで作成できる。インターネットブラウザ42は、ブラウジング操作によりサーバ用コンピュータ32のWWWサーバ51に接続し、WWWサーバ51に保持されているHTMLで作成されたファイル(HTMLファイル59)をダウンロードしインターネットブラウザ42の画面にワークフロー定義受付ページを表示する。このワークフロー定義受付ページから、ユーザが所定操作を行うことで、作成済みのWPDL-XMLファイル43を電子メールに添付し、WWWサーバ51に転送する。

【0035】サーバ用コンピュータ32は、例えばUNIX(登録商標)、WindowsNT(登録商標)等のOSをインストールしたコンピュータであり、WWWサーバ51、WWWサーバ側プログラム52、WPDL-XMLアダプタ53、56、ワークフローサーバ35用のクライアントライブラリ55、ワークフローサーバ38用のクライアントライブラリ58等を有している。

【0036】WWWサーバ51は、HTTPリクエストを処理するサーバである。WWWサーバ側プログラム52は、WWWサーバ51へのリクエストに連動するように設定されたサーバ側プログラムである。サーバ側プログラムとしては、WWWサーバ51のCGI(Common Gateway Interface)機能を使用しても良く、また、WWWサーバ51に対応した市販のアプリケーションサーバ製品を使用して組み込んでも良い。いずれかの方法を用いることが一般的である。市販のアプリケーションサーバ製品としては、Java(登録商標)Servlet API対応、Enterprise Java BeansAPI対応、CORBA対応などの様々な製品があり、必要とする機能によって製品を選定する。例えばWebsphere Application Server(IBM社製)、JRun Server(Allaire社製)、WebLogic Application Server(BEA社製)等、様々な製品が使用可能である。

【0037】WWWサーバ側プログラム52は、クライアントPC31から転送されてきたWPDL-XMLファイル43を、サーバ用コンピュータ32上の所定のディレクトリに格納し、サーバ用コンピュータ32上に存在する該当WPDL-XMLアダプタ53、56の登録処理を起動する。WPDL-XMLファイル43は、クライアントPC31から転送されたワークフロー定義情報である。

【0038】WPDL-DTDファイル60は、サーバ用コンピュータ32に格納される文書型定義情報である。このWPDL-DTDファイル60は、このようにインターネット/イントラネット上の所定のサーバに格納しておき、URL(Un

iform Resource Locator)形式でクライアントPC31から参照するようにしても良く、また、個々のクライアントPC31に格納しても良い。

【0039】WPD-XMLアダプタ53は、ワークフローシステム11用のWPD-XMLアダプタであり、XMLパーサ54を有している。XMLパーサ54は、WPD-XMLアダプタ53の内部で、WPD-XMLファイル43の解析を行うプログラムであり、市販の製品を利用できる。

【0040】ワークフローシステム11用のクライアントライブラリ55は、ワークフローサーバ35が提供しているクライアント用ライブラリである。WPD-XMLアダプタ53は、このワークフローシステム11用のクライアントライブラリ55の機能を使用することにより、WPD-XMLデータをワークフローサーバ35に適合させる。

【0041】ワークフローサーバ35は、ワークフロー処理を実行するサーバプログラムであり、ワークフローシステム11用のクライアントライブラリ55からのリクエストを受け付けてこのワークフローサーバ35独自の方式でデータを登録する。

【0042】WPD-XMLアダプタ56は、ワークフローシステム21用のWPD-XMLアダプタであり、XMLパーサ57を有している。XMLパーサ57は、WPD-XMLアダプタ56の内部で、WPD-XMLファイル43の解析を行うプログラムであり、市販の製品を利用できる。

【0043】ワークフローシステム21用のクライアントライブラリ58は、ワークフローサーバ38が提供しているクライアント用ライブラリである。WPD-XMLアダプタ56は、このワークフローシステム21用のクライアントライブラリ58の機能を使用することにより、WPD-XMLデータをワークフローサーバ38に適合させる。

【0044】ワークフローサーバ38は、ワークフロー処理を実行するサーバプログラムであり、ワークフローシステム21用のクライアントライブラリ58からのリクエストを受け付けて、このワークフローサーバ38独自の方式でデータを登録する。

【0045】図3は、図1の論理モデルに従い、サーバあるいはクライアントのコンピュータに実装したプログラムの一例を示す図である。

【0046】ワークフローシステムのプログラムとしては、WPD-XML InConcertアダプタ71、InConcert Java連携ライブラリ72、InConcertサーバ73などである。

【0047】WPD-XML InConcertアダプタ71は、WPD-XMLのワークフロー定義データを、InConcertのワークフローデータとして登録し、また、InConcertのワークフローデータを、WPD-XML形式で出力する、双方向変換アダプタである。このWPD-XML InConcertアダプタ71には、IBM XML Parser for Java74が使用されている。IBM XML Parser for Java74は、XMLパーサ/レンダラである。

【0048】InConcert Java連携ライブラリ72は、Java言語から利用できるInConcertクライアントライブラリである。InConcertサーバ73は、ワークフロー管理システムInConcertのサーバプログラムである。

【0049】クライアント側のアプリケーションとしては、WPD-XMLワークフロー定義プログラム75、WPD-XMLワークフロー表示プログラム76等である。各プログラムには、IBM XML Parser for Java74が使用されている。

【0050】プログラム言語にはJavaが使用されている。WPD-DTDファイル75は、WPDの文法を基にXMLでの表記法を決め、タグを定義したものである。WPD-XMLワークフロー定義プログラム75は、ユーザがWPD-XMLの表記法を知らなくても、GUI操作（アイコン部品のドラッグアンドドロップ操作）によりワークフロー定義が行え、GUI操作に応じてWPD-XMLファイルを自動生成する定義プログラムである。

【0051】WPD-XMLワークフロー表示プログラム76は、WPD-XMLのワークフロー情報をグラフィカルに表示するプログラムである。これらのプログラムは、Javaのアプリレットで開発されているので、インターネットブラウザを利用してWWWサーバ上にあるWPD-XMLファイルを表示することができる。

【0052】ここで、図4を参照し上記図3の実装プログラムをインターネットのシステムモデルとして実現した例について説明する。図4に示すように、サーバ用コンピュータ80には、Java Servlet対応アプリケーションサーバ81、WWWサーバ51、WPD-XML InConcertアダプタ71、InConcert Java連携ライブラリ72等を実装する。Java Servlet対応アプリケーションサーバ81は、Java Servlet82を有している。

【0053】クライアントPC90には、HTMLファイルやWPD-XMLファイルをダウンロードするインターネットブラウザ42、WPD-XMLワークフロー定義プログラム75、WPD-XMLワークフロー表示プログラム76等を実装する。

【0054】また、各コンピュータ80、90には、WPD-DTDファイル75を格納しておく。

【0055】ワークフロー処理を実行するワークフロー処理用コンピュータ100には、InConcertサーバ73とワークフロー定義データを格納したデータベース101を実装する。

【0056】以下、個別の実装例について説明する。まず、WPDの表記法をXMLで記述するための表記法について説明する。WPDでは、ワークフローの用語及び構造をWPDワークフローモデルとして定義している。このワークフローモデルを記述するための文法を、WPD表記法と呼ぶ。本発明では、ワークフローモデルをそのままに、XMLでワークフローモデルを記述できるように表記法を変更した。この表記法をWPD-XML表記法と呼ぶことにす

る。

【0057】WPDL表記法では、ワークフロー情報を以下のように表記する。

キーワード '識別子'

サブキーワード "値の文字列" または値の予約語または '識別子の参照'

(例1)

PARTICIPANT	'Sales_Department'
NAME	"Sales Department"
DESCRIPTION	"is involved in handling sales orders"
TYPE	ORGANISATIONAL_UNIT
END_PARTICIPANT	

【0059】この(例1)の表記を、以下のルールで、XMLで表記する。

・キーワードは、同名のXMLエレメント(タグ)とする。

・キーワードの識別子は、ID属性とする。

・サブキーワードは、同名のXMLエレメントとする。

・予約語は、同名のXMLエレメントとする。

・サブキーワードの値が、文字列の場合、サブキーワードと同名のXMLエレメントの内容とする(ダブルクォーテーションでは囲まない)。

・サブキーワードの値が、予約語の場合、予約語と同名※

(例2)

```
<PARTICIPANT ID="Sales_Department">
  <NAME>Sales Department</NAME>
  <DESCRIPTION>is involved in handling sales orders</DESCRIPTION>
  <TYPE><ORGANIZATIONAL_UNIT/></TYPE>
</PARTICIPANT>
```

【0061】例外ルールとして、下記のものを設定する。

・キーワードACTIVITYのサブキーワードIMPLEMENTATION 30の値となる予約語、NONE、APPLICATIONS、WORKFLOW、LOOPは、XMLエレメントとせず、IMPLEMENTATIONエレメントのTYPE属性として表記する。

・キーワードACTIVITYのサブキーワードSPLITの値とな

*...

END_キーワード

具体的には、下記(例1)のように表記される。

【0058】

【数1】

*

※のXMLエレメントを指定する。

・サブキーワードの値が、識別子の参照であり、かつ一つだけの場合、サブキーワードのIDREF属性の値とする。

・サブキーワードの値が、識別子の参照であり、かつ複数ある場合、サブキーワードのIDREFS属性の値とする。この場合、上記(例1)のWPDLの表記は、下記(例2)のようにXMLで表記できる。

【0060】

【数2】

る予約語、AND、OR、XORは、XMLエレメントとせず、SPLITエレメントのTYPE属性として表記する。以下に、これらの例外を含んだ元のWPDL表記例(例3)と、WPDL-XMLの表記例(例4)を示す。

【0062】

【数3】

13
(913)
ACTIVITY
NAME "Mail Room"
IMPLEMENTATION APPLICATIONS
TOOL_LIST
"scan_document"
"identify_document"
"send_document"
DESCRIPTION "sequential execution"
END_TOOL_LIST
PERFORMER "Mail_Room_Clerk"
SPLIT AND
END_ACTIVITY

14

(914)
<ACTIVITY ID="MailRoom">
<NAME>Mail Room</NAME>
<IMPLEMENTATION TYPE="APPLICATIONS">
<TOOL_LIST IDREFS="scan_document identify_document send_document">
<DESCRIPTION>sequential execution</DESCRIPTION>
</TOOL_LIST>
</IMPLEMENTATION>
<PERFORMER IDREF="Mail_Room_Clerk"/>
<SPLIT TYPE="AND" />
</ACTIVITY>

【0063】続いて、WPDL-XMLワークフロー定義プログラム75（ワークフロー定義プログラム）とWPDL-XMLワークフロー表示プログラム76（ワークフロー表示プログラム）について説明する。WPDL-XMLは、前述のWPDL-X
ML表記法を理解することにより、Windowsのメモ帳のよ
うなテキストエディタ（テキスト編集ソフト）で作成す
ることができるが、これでは、WPDL-XML表記法を理解し
たものでなければ、WPDL-XMLを記述することができな
い。

【0064】そこで、本発明では、ワークフロー定義プ
ログラムとしてWPDL-XMLワークフロー定義プログラム7
5を設けた。

【0065】WPDL-XMLワークフロー定義プログラム75
は、図5に示すように、XMLパーサ/レンダラ110、デ
ータオブジェクト生成部112、ワークフロー定義画面
表示部114、XML表示画面表示部116、変換処理部
118等からなる。

【0066】XMLパーサ/レンダラ110は、WPDL-XMLフ
ァイルを読み込み解析する。また、XMLパーサ/レンダラ
110は、データオブジェクト生成部112で生成され
た内容（データオブジェクトクラスのインスタンス）
に応じてXMLエレメントオブジェクトを作成する。さら
に、XMLパーサ/レンダラ110は、ワークフロー定義画
面20
面で定義された内容を、WPDL-DTDファイルに基づきWPDL
-XMLファイルとして書き出す。データオブジェクト生成
部112は、解析されたデータから、Javaで利用するた
めのデータオブジェクトクラスのインスタンスを生成す
る。ワークフロー定義画面表示部114は、ワークフロ
ー定義画面を表示する。

【0067】WPDL-XMLワークフロー定義プログラム75
及びWPDL-XMLワークフロー表示プログラム76では、デ
ータオブジェクトクラスのインスタンスのデータ構造が
アイコンや矢印等で表示される。XML表示画面表示部1
50

16は、XMLパーサ/レンダラ110により変換（作成）
されたXML文字列を表示する。変換処理部118は、XML
パーサ/レンダラ110により変換（作成）されたXML文
字列をWPDLのファイルに変換する。

【0068】クライアントPC31にインストールした
WPDL-XMLワークフロー定義プログラム75を起動するこ
とで、図6に示すようなワークフロー定義画面が表示さ
れる。このワークフロー定義画面は、オブジェクトを個
々に定義する定義画面12.1と、Flowタブの画面12.2
とXMLタブの画面12.3等の2つを切替え表示する編集
用画面とから構成されている。

【0069】起動時は、ワークフロー定義画面には、Fl
owタブの画面12.2が表示されており、ツールバー部分
に表示されたアイコンの中から例えばコンピュータのアイ
コン12.5等をユーザが選択し編集用画面12.2上に
移動及び配置する操作（ドラッグアンドドロップ等の操
作）を行い、この操作を繰り返し、さらに、ツールバー
部分の矢印アイコン12.6で編集画面上のコンピュータ
のアイコン12.5を接続することによって、WPDLワーク
フローモデルに従ったワークフローが定義される。ま
た、個々のコンピュータのアイコン12.5に対して定義
画面12.1にて値（業務種別、名称、役職、役柄等）を
設定することで、WPDL-XMLが自動作成される。

【0070】ここで、ユーザがワークフロー定義画面の
XMLタブを選択操作すると、図7に示すように、WPDL
ワークフローモデルに従ったワークフロー定義をXMLで
記述したテキストデータ（XML文字列）がXMLタブの画面
12.3に表示される。このようなXML文字列は、ワーク
フロー定義の途中でも確認することができる。

【0071】続いて、図8を参照してWPDL-XMLワークフ
ロー定義プログラム75によるWPDL-XMLファイル編集処
理動作について説明する。ワークフロー定義画面にて、
解析対象のWPDL-XMLファイルが指定されると、WPDL-XML

ワークフロー定義プログラム75では、指定されたWPDL-XMLファイルがXMLパーサ/レンダラ110によって読み込まれ、読み込まれたWPDL-XMLデータが解析される。

【0072】解析後、WPDL-XMLワークフロー定義プログラム75では、その解析したWPDL-XMLデータは、修正、追加等の変更が可能のように、データオブジェクトクラスのインスタンスとして作成されて(図8のS1001)、データオブジェクト生成部112に保持される(S1002)。データオブジェクトクラスは、XMLエレメント(タグ)の名前に対応して予め作成されている。

【0073】データオブジェクト生成部112に保持されたデータオブジェクトクラスのインスタンスは、ワークフロー定義画面表示部114に出力されてワークフロー定義画面上に、処理の流れを示すデータオブジェクトとして表示される(S1003)。

【0074】ここで、ユーザにより画面操作が行われると(S1004)、その操作内容に沿ってワークフロー定義が編集されて、データオブジェクトクラスのインスタンスとしてデータオブジェクト生成部112に保持される(S1002)。

【0075】データオブジェクト部112のデータオブジェクトクラスのインスタンスは、W3Cの定義するXMLのエレメントオブジェクトクラスのインスタンスを生成し、XMLパーサ/レンダラ110に渡す。

【0076】XMLパーサ/レンダラ110では、XMLレンダラ機能を使って、XMLのエレメントオブジェクトクラスのインスタンスを、WPDL-XMLファイルとして書き出す(S1006)。

【0077】この書き出し時に、WPDL-XMLファイルを変換処理部118に出力し、変換処理部118でフォーマットをWPDLの形式に変換することにより、WPDLの表示形式で書き出すようにもできる。

【0078】なお、本実施形態では、XMLの解析及び生成のために、IBM XML Parser for Javaを使用したのが、この製品は、W3C(World Wide Web Consortium)が標準化した、XML Java APIに準拠した製品である。したがって、XML Java APIに準拠した製品であれば、これ以外の製品を使用することもできる。

【0079】すなわち、WPDL-XMLワークフロー定義プログラム75をユーザが使用することで、ユーザは、WPDLワークフローモデルについての知識さえあれば、WPDLの表記法もWPDL-XMLの表記法も知らなくても、WPDL-XMLを作成することができる。

【0080】次に、WPDL-XMLワークフロー表示プログラム76について説明する。このWPDL-XMLワークフロー表示プログラム76は、図9に示すように、XMLパーサ130、データオブジェクト生成部132、ワークフロー表示画面表示部134等からなる。XMLパーサ130は、最新のWPDL-XMLファイルを読み込み解析する。デー

タオブジェクト部132は、画面表示のためのデータオブジェクトを保持する。ワークフロー表示画面表示部134は、実行中のワークフローの進捗状況を表示する。

【0081】続いて、このWPDL-XMLワークフロー表示プログラム76のワークフロー進捗状況表示動作について説明する。なお、処理動作そのものは図8のS1001～S1003までと同じである。

【0082】クライアントPC90のインターネットブラウザの画面にて、ユーザによって、ワークフロー処理が実行されているシステムのワークフロー進捗状況情報を示すWPDL-XMLファイルが指定されると、WPDL-XMLワークフロー表示プログラム76が起動し、指定されたWPDL-XMLファイルがXMLパーサ130によって、ワークフローサーバ側からインターネットを通じて読み込まれてそのWPDL-XMLデータが解析される。この解析にはWPDL-DTDファイルが利用される。

【0083】解析後、そのWPDL-XMLデータは、データオブジェクトクラスのインスタンスとされて(図8のS1001)、データオブジェクト生成部132に保持される(S1002)。データオブジェクトクラス自体は、XMLエレメント(タグ)の名前に対応して予め作成されている。

【0084】データオブジェクト生成部132に保持されたデータオブジェクトクラスのインスタンスは、ワークフロー定義画面表示部134に出力されて、図10に示すようなワークフロー表示画面(Flow Viewer)上に、現在のワークフローの進捗状況を示すデータオブジェクトとして表示される(S1003)。

【0085】この例では、パソコンの購入申請のワークフローにおいて、ある書面は上長承認を却下されて再申請指示のボックスに入れられ、再申請指示待ちの状態になっていることが解る。また、他の書面は、上長承認、購入決裁、購買処理を通過し、登録のボックスと通知のボックスにそれぞれ入れられていることが解る。

【0086】続いて、WPDL-XML InConcertアダプタ71について説明する。WPDL-XML InConcertアダプタ71は、XMLパーサ/レンダラ141、データオブジェクト生成部143、データ登録部145、データ取得部147等からなる。データオブジェクト生成部143は、Javaプログラムで生成されている。XMLパーサ/レンダラ141としては、WPDL-XMLの解析及び生成のために、ワークフロー定義プログラム75と同様に、IBM XML Parser for Javaが使用されている。データオブジェクト生成部143は、XMLパーサ/レンダラ141により解析された情報を、データ登録部145で使うためのデータオブジェクトクラスのインスタンスに変換する。データ登録部145は、WPDLのワークフローモデルに従ったデータを、InConcertのワークフローモデルに変換してInConcertに登録する。データ取得部147は、InConcertから取得したデータを、WPDLのワークフローモデルに従

ったデータ形式に変換する。

【0087】なお、InConcert Java連携ライブラリ72は、InConcertのクライアント機能をJavaプログラムから利用するためのクラスライブラリである。

【0088】続いて、このWPD-XML InConcertアダプタ71の動作を説明する。この場合、変換対象のWPD-XMLファイルが指定されると、XMLパーサ/レンダラ141は、指定された変換対象のWPD-XMLファイルを読み込み、解析する。そして、解析した情報をデータオブジェクト生成部143に渡す。

【0089】データオブジェクト生成部143は、XMLパーサ/レンダラ141によって解析された情報が渡されると、その情報をデータ登録部145で使うためのデータオブジェクトクラスのインスタンスに変換しデータ登録部145に渡す。

【0090】データ登録部145は、データオブジェクト生成部143よりデータオブジェクトクラスのインスタンスを受け取ると、InConcert Java連携ライブラリ72を参照し、WPDのワークフローモデルに従ったデータを、InConcertのワークフローモデルに変換してInConcertに登録する。

【0091】一方、図4に示したシステム構成において、InConcertサーバ73におけるワークフローの進捗状況をクライアントPC90のユーザが確認するために、インターネットブラウザ42からWWWサーバ51のワークフロー定義XML受付ページにアクセスし、表示されたワークフロー定義XML受付ページにて所定の操作を行うと、Java Servlet対応アプリケーションサーバ81によってWPD-XML InConcertアダプタ71が起動される。

【0092】すると、WPD-XML InConcertアダプタ71では、図11に示したデータ取得部147がInConcertサーバ73からデータを取得し、取得したデータをデータオブジェクト生成部143に渡す。

【0093】データオブジェクト生成部143では、渡されたInConcertのデータをWPDのワークフローモデルに従ったデータ形式に変換し、XMLパーサ/レンダラ141に渡す。

【0094】XMLパーサ/レンダラ141は、データオブジェクト生成部143より渡されたWPDのデータをXMLに変換してWPD-XMLファイルとして出力する。

【0095】すなわち、WPD-XML InConcertアダプタ71は、ワークフローシステムInConcertを対象に、WPD-XMLのワークフロー定義情報を、InConcertのワークフロー定義情報として登録し、InConcertのワークフロー定義情報を、WPD-XML形式で出力するものである。

【0096】なお、WPDのワークフローモデルは、既存のワークフローシステムが個別に使用しているワークフローモデルと同一とは限らない。用語や構造になんらかの差異があり得る。したがって、上記以外は、WPD-XML

InConcertアダプタ71ではなく変換アダプタとなり、WPDのワークフローモデルを、個別のワークフローシステムのモデルに変換し、また、その逆を行うプログラムとする。

【0097】InConcertとWPDとは、構造的に共通点が多く、主な変換は用語の違いと言える。したがって、変換プログラムは、比較的容易に開発できる。しかし、実行者の指定法、条件の指定法、サブフローの指定法など、構造が異なる部分もある。

【0098】WPDのワークフローモデルとInConcertのワークフローモデルとの間の差異と、その変換方法について説明する。

【0099】双方向変換するにあたり、図12に示すような事項を考慮する必要がある。

【0100】この中でも、特に、考慮を要した4つの点について説明する。1. 実行者の指定WPDとInConcertの実行者指定の方法には、図13のような違いがある。

【0101】例えばWPDでは、作業単位であるACTIVITYのPERFORMER属性に対し、HUMAN（人）、ORGANIZATIONAL UNIT（部門）、ROLE（役割）、SYSTEM（アプリケーション）のいずれかのPARTICIPANTオブジェクトを割り当てるが、InConcertの場合には、作業単位Taskに対し、Role（役割）オブジェクトを割り当て、Roleの実行者として、Poolオブジェクトを割り当てる。PoolオブジェクトはUser（利用者）の集合である。

【0102】この2つの異なるモデルを、以下のように変換する。

【0103】（WPD→InConcert）

・HUMAN： HUMANエレメントは人を表すので、Userに相当する。しかしInConcertではUserを含むPoolを作成しなければならないので、UserとUserを含むPoolの両方を作成する。

・ORGANIZATIONAL_UNIT： ORGANIZATIONAL_UNITエレメントは部門を表す。InConcertではPoolがこれに相当する。ORGANIZATIONAL_UNITに所属する利用者のリストが定義時に得られれば、その数だけUserを作成し、Poolに割り当てる。

・ROLE： ROLEエレメントは役割を表す。Roleオブジェクトを作成して、Taskに割り当てる。

・SYSTEM： SYSTEMエレメントはアプリケーションの実行を表す。InConcertでは、Taskに直接アプリケーションを指定することはできないが、Taskの開始時のイベントをキャッチして特定のアプリケーションに起動をかける設定をすることができる。これを利用し、SYSTEMエレメントで指定されたアプリケーションを起動するAction Specオブジェクトを作成。次に、タスク開始イベントから、これを実行するTriggerオブジェクトを作成する。

【0104】（InConcert→WPD）

・Pool： InConcertのPoolオブジェクトは、HUMANエレメントに相当する場合と、ORGANIZATIONAL_UNITエレメ

ントに相当する場合が考えられる。この判別方法として、Poolに所属するUserが単一のときはHUMANエレメント、複数のときはORGANIZATIONAL_UNITエレメントを作成する。

・Role: ROLEエレメントを作成する。

・Trigger: InConcertのTaskにTriggerが設定されている場合は、起動されるActionSpecオブジェクトから、アプリケーション名を取得し、SYSTEMエレメントを作成する。

【0105】2. 条件の指定

図17に示すように、ワークフローでの条件は、ある作業単位から別の作業単位に遷移するときに設定される。WPDLでの条件は、分岐条件であり、ある作業単位の次のルートが複数ある場合、条件判断を行って、通過するルートを決める。条件に合致しなかったルートは、通らない。InConcertでの条件は、ルートを選択する条件ではなく、作業の実行条件である。存在するルートは、無条件に通過し、作業単位に到達してから、条件判断を行い、作業単位を実行するかどうかを決める。

【0106】変換時には、それぞれの実行条件を、拡張属性として保持する。

【0107】・WPDL→InConcert WPDL_CONDITION属性の値として保持する。

・InConcert→WPDL EXTENDED_ATTRIBUTEエレメントのPERFORM_CONDITIONの値として保持する。

【0108】3. サブフローの指定

WPDLもInConcertも、ワークフローの作業の一部として、別のワークフローを指定することができる。

【0109】例えば図15に示すように、WPDLでは、並列した複数のワークフローを記述して、一方のワークフローの作業単位(ACTIVITY)で、もう一方のサブフローとして指定する。InConcertでは、作業単位(Task)を階層構造にすることにより、サブフローの指定が可能である。

【0110】4. 拡張属性の扱い

WPDLでは、ワークフロープロセスに任意の拡張属性を追加できる。これをInConcertに登録するときに以下の処理を行う。

【0111】WPDL情報を登録するWPDLクラスに、属性名を格納する属性と、属性値を格納する属性を予め用意する。この組を、整数型、文字列型、日付型の3種類用意する。つまり、以下の6通りである。

【0112】・整数属性名を格納する属性(文字列型)

・整数属性値を格納する属性(整数型)

・文字列属性名を格納する属性(文字列型)

・文字列属性値を格納する属性(文字列型)

・日付型属性名を格納する属性(文字列型)

・日付型属性値を格納する属性(日付型)

複数の属性を登録できるように、この6通りをそれぞれ複数個用意しておく。WPDL-XMLにより与えられた属性が

その数を上回ったときはその都度属性の組を追加する。【0113】ここで、図16を参照してWPDL-XMLの拡張属性をInConcertに登録する場合の一例について説明する。

【0114】WPDL-XMLでは、ワークフロープロセスに任意の拡張子を追加できるが、これをそのままInConcertに登録すると、InConcert側では、変換のたびにデータベース上に新しい属性を追加する処理を実行することになりこの際の負荷が重くなる。

【0115】そこで、本実施形態の場合、図16に示すように、InConcert側に予め属性名と属性値を格納するための属性(テンプレート)151を用意しておく。

【0116】例えば登録対象のWPDL-XML150が、図のような名前(STR_A, STR_B, INT_A)、値("AA", "BB", 5)、型(String, String, Integer)を持っている場合、この属性(テンプレート)151を使用することによって、行を1行ずつ追加しながら、それぞれの名前、値、型を当てはめるだけで良くなり、InConcert側で変換のたびにデータベース上に新しい属性を追加する処理の負荷を軽減することができる。

【0117】続いて、独立したワークフローシステム間の連携モデルについて説明する。図2で示した、インターネットにおけるシステムモデルは、一つのWWWサーバの管理下に、複数の異なるワークフローシステムが存在するような場合、例えば、企業のメインのホームページのバックエンドに、部門ごとに異なるワークフローサーバが動作している場合や、グループ企業のメインのホームページのバックエンドに、企業ごとに異なるワークフローサーバが動作している場合などに有効である。

【0118】これに対し、システムを共有しない企業間の異なるワークフローシステムを連携させたい場合がある。この場合、システム間をつなぐ通信方式(メール、メッセージング、分散オブジェクトなど)が必要になる。

【0119】複数のワークフローサーバのいずれか、またはすべてが、WWWサーバの管理下でない場合は、図2の構成を一部拡張し、図17に示すような構成をとる必要がある。この例では、ワークフローシステムが、WWWサーバ51の管理下にはない独立したシステムであるような場合を想定している。

【0120】すなわち、この例の場合、サーバ用コンピュータ32に、通信プログラム83を新たに設ける一方、WWWサーバ51の管理下にはない独立したワークフローシステム160に通信プログラム161、WPDL-XMLアダプタ57及びワークフローサーバ38用のクライアントライブラリ58を設ける。

【0121】通信プログラム83は、ワークフローシステム160に対し、WPDL-XMLデータの転送を行う通信プログラムである。通信プログラム83の実装形態としては、分散オブジェクト、メッセージングシステム、メー

ル等、いくつかの方法が考えられる。

【0122】例えば分散オブジェクトの場合は、CORBAなどの分散オブジェクト製品を使用してWPD-LXMLデータを転送する。メッセージングシステムの場合は、非同期の通信が可能なメッセージング製品を使用してWPD-LXMLデータを転送する。メールの場合は、メールの添付ファイルを使用してWPD-LXMLデータを転送する。

【0123】通信プログラム（受信側）161は、転送されてきたWPD-LXMLデータを所定のディレクトリにファイルとして書き出す。WPD-LXMLファイル43は、転送されてきたワークフロー定義情報ファイルである。WPD-LXMLファイル60は、ワークフローシステム160とそのコンピュータ37に配置されている。なお、DTDファイルは各コンピュータ37に配置せず、インターネット上の特定に位置に配置し、URL形式で参照するようにしても良い。WPD-LXMLアダプタ57は、ワークフローシステム160のためのWPD-LXMLアダプタである。XMLパーサ54は、WPD-LXMLアダプタ57の内部で、WPD-LXMLファイルの解析に使用されるプログラムである。これは、市販のXMLパーサ製品を利用できる。

【0124】ワークフローシステム160用のクライアントライブラリ58は、ワークフローシステム160が提供しているクライアント用ライブラリである。WPD-LXMLアダプタ57は、このライブラリの機能を使用することにより、WPD-LXMLデータをワークフローシステム160に適合させる。

【0125】ワークフローシステム160用のサーバ38は、ワークフローシステム160のサーバプログラムである。リクエストを受け付けて、製品独自の方式でデータを登録する。

【0126】このワークフロー連携システムの場合、サーバ用コンピュータ32の通信プログラム83と、独立したワークフローシステム160の通信プログラム161間でWPD-LXMLファイルのやりとり（転送）を行うことで、独立したワークフローシステム160であってもワークフローXML、XMLワークフロー等の双方向変換が可能になり、ワークフロー連携を行うことができる。

【0127】ここで、WPD-LXMLの表記法を、XMLで記述するための表記法の別の表記ルールについて説明する。

【0128】今回の実施形態は、WPD-LXMLの表記法から、極力少ないルールで、XMLの表記法に変換することを目的としている。したがって、ごくわずかの例外をのぞいて、すべてのWPD-LXMLキーワードやサブキーワードはXML要素（タグ）として定義した。このことにより、変換作業を単純な作業にすることができる。

【0129】しかし、WPD-LXMLのワークフローモデルの意味を詳細に調査すると、サブキーワードのいくつかはXML要素ではなく、むしろ要素の属性として定義した方が適当な場合があることも事実である。

【0130】例えば個々の仕事(Activity)の名前は、現在<NAME>タグによって記述される。

<ACTIVITY>

<NAME>申請書提出</NAME>

</ACTIVITY>しかし、一個の要素に必ず一つ現れるようなデータは、属性で定義した方が自然である。

【0131】<ACTIVITY NAME=" 申請書提出" />このようにWPD-LXMLワークフローモデルの意味を重視して、要素と属性を使い分ける表記法も考えられる。

【0132】WPD-LXMLワークフロー定義プログラム、WPD-LXMLワークフロー表示プログラム、WPD-LXML InConcept変換アダプタ等の各プログラムは、本実施形態において、Java言語によって作成されているが、この他、例えばWindows上のVisualC++、VisualBasic等でも作成できる。

【0133】このようにこの実施形態のワークフロー連携システムによれば、汎用的なワークフロー定義であるWPD-LXMLをXMLで記述したWPD-LXMLファイルを、インターネットあるいはイントラネットを介して接続された異なるワークフローシステム間でやりとりすることでワークフロー連携を容易に行うことができる。

【0134】また、WPD-LXMLの文法をXMLの表記法で忠実に再定義し、文書型定義を記述したDTDファイルを作成したので、WPD-LXMLをXMLで表記することが可能になった。これにより、市場に広く出回っている任意のXMLパーサ(XML文書構造を解析するプログラム)を使用して言語処理を行えるようになり、XMLを使わない場合と比較して開発の工数を削減することができる。

【0135】また、ユーザは、ワークフロー定義画面に表示されたアイコンをドラッグアンドドロップするというGUI操作でワークフローモデルを作成（表記）できる。しかも、そのとき作成したワークフローモデルがXMLの表記に自動的に変換されるので、ユーザは、WPD-LXMLのワークフローモデルについての知識さえあれば、WPD-LXMLの表記法もXMLの表記法も知らなくても、テキストエディタ等でキー入力あるいは編集操作することなくXMLで表記したワークフローモデルを容易に作成することができる。

【0136】さらに、XMLによるWPD-LXML対応をそれぞれのワークフローシステムが行えば、異なるワークフローシステムが共通の定義データとアプリケーションプログラムを利用できるようになる。

【0137】なお、本発明は上記実施形態のみに限定されるものではない。上記実施形態に記載の各プログラム（ソフトウェア）は、フロッピー（登録商標）ディスクなどのコンピュータが読み出し可能な記憶媒体に記憶されていても良く、この場合、記憶媒体に記憶されたプログラム（ソフトウェア）をコンピュータが読み出すことにより、各実施形態における処理が可能になる。

【0138】なお、本発明における記憶媒体としては、

磁気ディスク、フロッピーディスク、ハードディスク、光ディスク（CD-ROM、CD-R、DVDなど）、光磁気ディスク（MOなど）、半導体メモリなど、プログラムを記憶でき、かつコンピュータが読み取り可能な記憶媒体であれば、その記憶形式はいずれの形態であっても良い。

【0139】また、記憶媒体からコンピュータにインストールされたプログラムの指示に基づき、コンピュータ上で稼動しているOS（オペレーティングシステム）や、データベース管理ソフト、ネットワークソフトなどのMW（ミドルウェア）などが本実施形態を実現するための各処理の一部を実行しても良い。

【0140】さらに、本発明における記憶媒体は、コンピュータと独立した媒体に限らず、LANやインターネットなどにより伝送されたプログラムをダウンロードして記憶または一時記憶した記憶媒体も含まれる。

【0141】また、記憶媒体は一つに限らず、複数の媒体から本実施形態における処理が実行される場合も本発明における記録媒体に含まれ、媒体構成はいずれの構成であっても良い。

【0142】なお、本発明におけるコンピュータは、記憶媒体に記憶されたプログラムに基づき、本実施形態における各処理を実行するものであって、パソコンなどの一つからなる装置、複数の装置がネットワーク接続されたシステムなどのいずれの構成であっても良い。

【0143】また、本発明におけるコンピュータとは、パソコンに限らず、情報処理機器に含まれる演算処理装置、マイコンなども含み、プログラムによって本発明の機能を実現することが可能な機器、装置を総称している。

【0144】

【発明の効果】以上説明したように、請求項1記載の発明によれば、ユーザは、画面上でのアイコン部品の配置操作、つまりドラッグアンドドロップの操作等でワークフローを定義でき、しかもこれと同時に、ワークフロー定義のXML変換が自動的に行われるので、ユーザは、あるワークフローモデルについての知識さえあれば、そのワークフロー定義の表記法もXMLの表記法も知らなくても、XMLで表記したワークフローモデルを作成することができる。

【0145】請求項2記載の発明によれば、ワークフロー処理が実行されているシステムより、ワークフローの進捗状況の情報をExtensible Markup Languageで表記したテキスト文字列で取得し、取得されたテキスト文字列を、予め規定された文書型定義に基づき、ワークフローの形態で表示するので、ユーザは、現在のワークフローの進捗状況をビジュアル的に確認することができる。

【0146】請求項3記載の発明によれば、ワークフローをExtensible Markup Languageで表記したテキスト文字列を解析し、ワークフロー連携ライブラリを参照する

ことで、他のワークフローモデルを生成し、生成されたワークフローモデルを該当ワークフローシステムに登録するので、異種ワークフロー間のワークフロー連携が可能になる。

【0147】請求項4記載の発明によれば、あるワークフローシステムのワークフローをExtensible Markup Languageで表記したテキスト文字列に変換し、この変換されたテキスト文字列を他のワークフローシステムへ伝送し、他のワークフローシステムでは、伝送されてきたテキスト文字列を、自身のワークフローに変換するので、一度、作成したワークフローモデルを他のワークフローシステムで再利用できるようになる。

【0148】請求項5記載の発明によれば、画面上に表示されたアイコン部品が画面上に配置されることでワークフローを定義し、このワークフローの定義をExtensible Markup Languageで表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Languageで表記したテキスト文字列に変換するので、テキストエディタ等を使わずとも、ワークフローの定義をExtensible Markup Languageで表記できる。

【0149】また、変換されたテキスト文字列を解析し、解析されたテキスト文字列を基に、所定言語のプログラムで編集可能のようにデータオブジェクトクラスのインスタンスを生成し、この生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成し、生成されたワークフローモデルを該当ワークフローシステムに登録するので、異種ワークフロー間のワークフロー連携が可能になる。

【0150】請求項6記載の発明によれば、クライアントコンピュータにおいて、あるワークフローシステムのワークフローをExtensible Markup Languageで表記したテキスト文字列に変換し、この変換されたテキスト文字列をネットワークを通じてサーバコンピュータへ伝送し、サーバコンピュータにおいてネットワークからテキスト文字列が受信された場合、変換対称ワークフローシステム用のワークフロー変換手段を起動し、起動されたワークフロー変換手段が、受信された文字列を、それぞれのワークフローシステム用のワークフローモデルに変換し、変換したワークフローモデルをそれぞれのワークフローシステムに登録するので、異なるワークフローシステム間のワークフロー同期をとることができる。

【0151】この結果、異なるワークフローシステム間で容易にワークフローを連携させることができる。

【図面の簡単な説明】

【図1】本発明に係る一つの実施の形態のワークフロー連携モデルを示す図。

【図2】図1のワークフロー連携モデルを具現化したワークフロー連携システムの構成例を示す図。

【図3】図1の論理モデルに従い、サーバあるいはクラ

イアントのコンピュータに実装したプログラムの一例を示す図。

【図4】図3のプログラムをインターネットのシステムモデルとした一例を示す図。

【図5】WPDL-XMLワークフロー定義プログラムの構成例を示す図。

【図6】ワークフロー定義画面の一例を示す図。

【図7】ワークフロー定義画面にて、ワークフロー定義をXMLで表示させた例を示す図。

【図8】ワークフロー定義処理及び表示処理を示すフローチャート。

【図9】WPDL-XMLワークフロー表示プログラムの構成例を示す図。

【図10】ワークフローの進捗状況を表示した一例を示す図。

【図11】WPDL-XML InConcertアダプタの構成を示す図。

【図12】WPDLのワークフローモデルとInConcertのワークフローモデルとの間の差異で変換に考慮する事項を示す図。

【図13】WPDLとInConcertとの実行者指定方法の違いを示す図。

【図14】WPDLとInConcertとの条件の指定方法の違い *

*を示す図。

【図15】WPDLとInConcertとのサブフローの指定方法の違いを示す図。

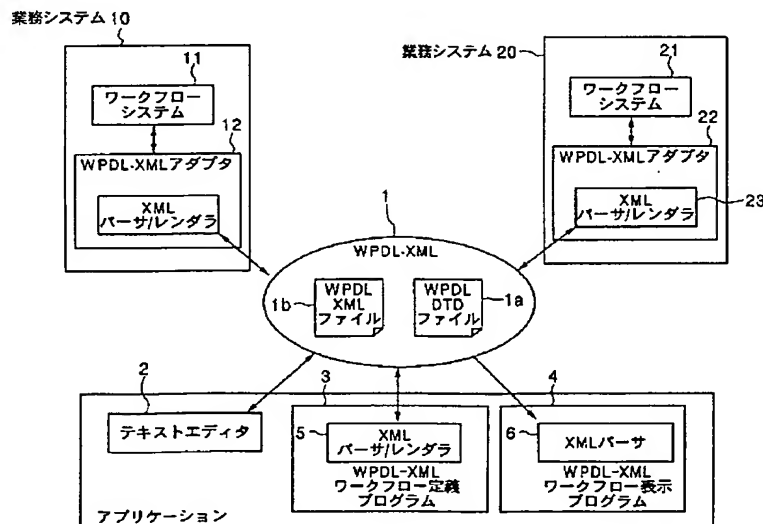
【図16】WPDL-XMLの拡張属性をInConcertに登録する場合の一例を示す図。

【図17】独立したワークフローシステム間の連携モデルを示す図。

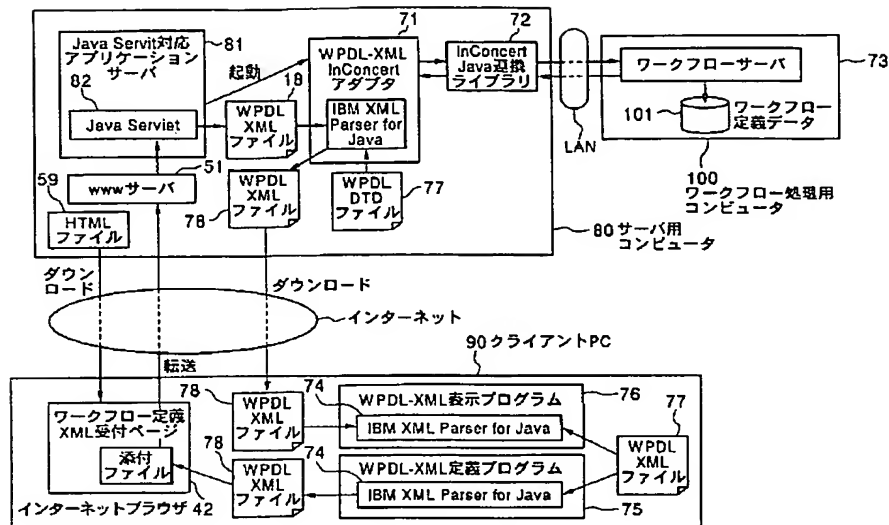
【符号の説明】

1…WPDL-XML、1a…WPDL-DTDファイル、1b…WPDL-XMLファイル、2…テキストエディタ、3…WPDL-XMLワークフロー定義プログラム、4…WPDL-XMLワークフロー表示プログラム、5、13、23…XMLパーサ/レンダラ、6…XMLパーサ、10、20…業務システム、11、21…ワークフローシステム、12、22…WPDL-XMLアダプタ、51…WWWサーバ、71…WPDL-XML InConcertアダプタ、72…InConcert Java連携ライブラリ、75…WPDL-XMLワークフロー定義プログラム、76…WPDL-XMLワークフロー表示プログラム、141…XMLパーサ/レンダラ、143…データオブジェクト生成部、145…データ登録部、147…データ取得部、160…ワークフローシステム、83、161…通信プログラム。

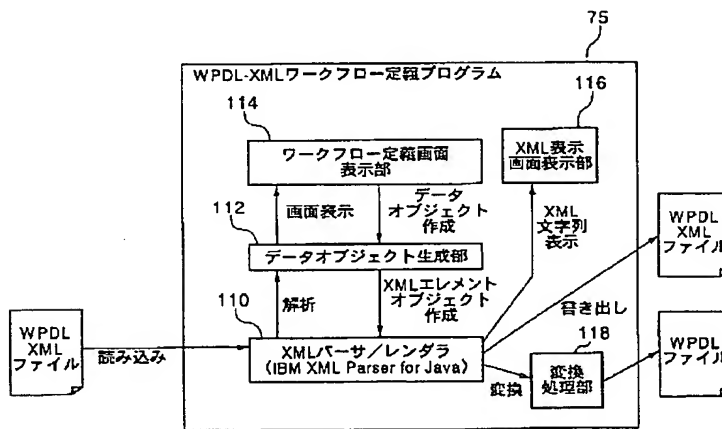
【図1】



【図4】



【図5】



【図16】

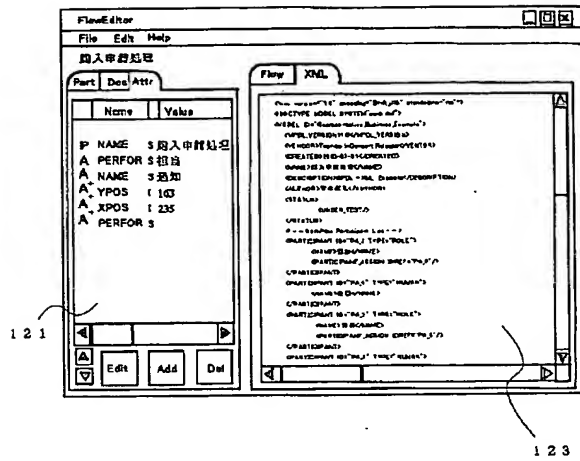
WPDML-XML ISO

名前	値	型
STR_A	"AA"	STRING
STR_B	"BB"	STRING
INT_A	5	INTEGER

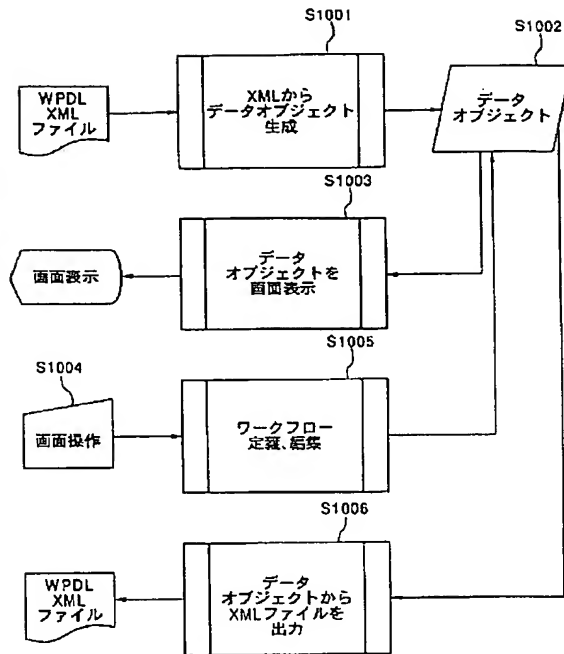
InConcert

名前	値	型
XROSS_S_NAME1	STR_A	STRING
XROSS_S_VALUE1	"AA"	STRING
XROSS_S_NAME2	STR_B	STRING
XROSS_S_VALUE2	"BB"	STRING
XROSS_I_NAME1	INT_A	STRING
XROSS_I_VALUE1	5	INTEGER

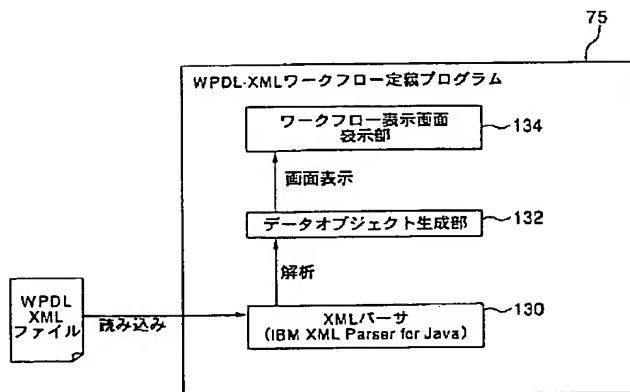
【図7】



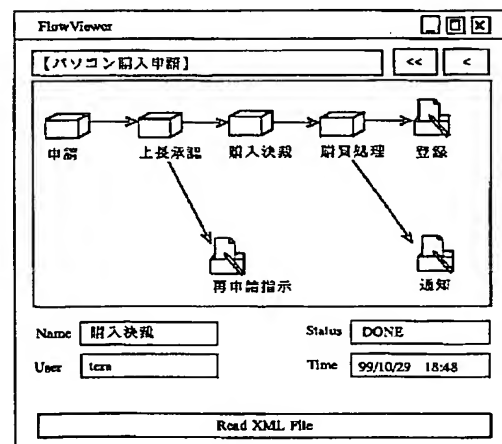
【図8】



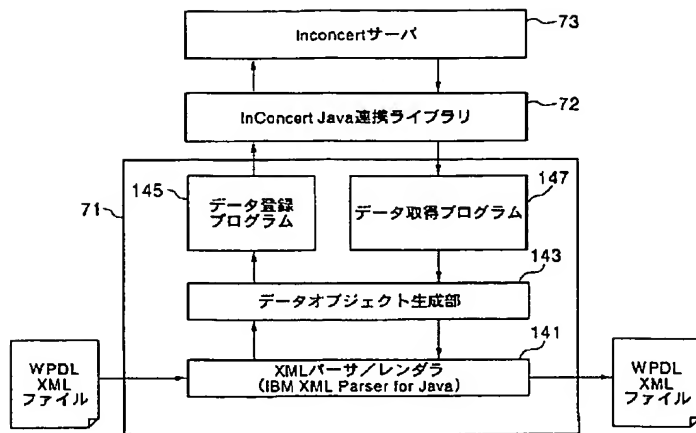
【図9】



【図10】



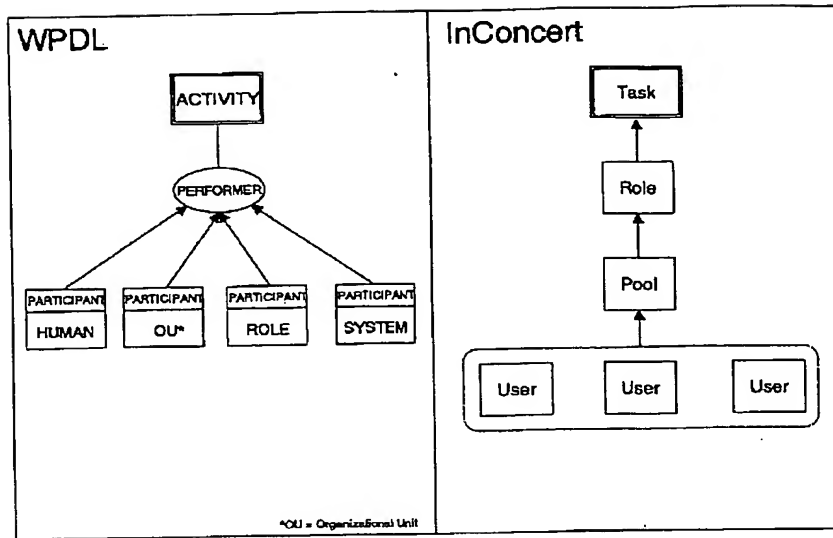
【図11】



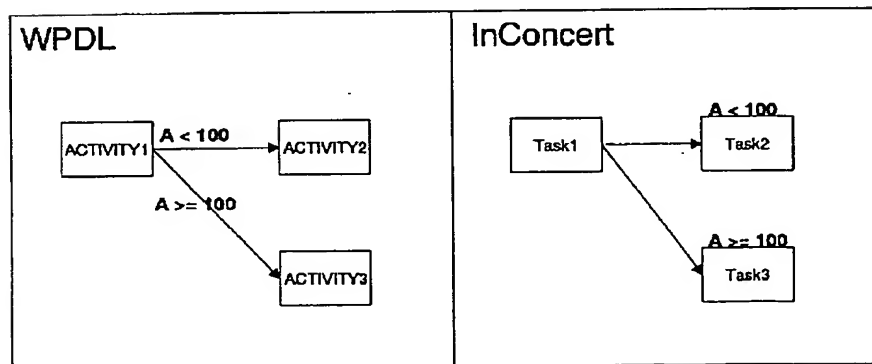
【図12】

用語/用法	WPDL-XML	InConcert	WPDL→InConcert	InConcert→WPDL
ワークフロー	WORKFLOW	Job	WPDL-XML情報を格納できるWPDLクラスを作成し、WPDLクラスを継承するJobテンプレートとして定義。	WORKFLOWエレメントを作成
個々の作業	ACTIVITY	Task	Taskオブジェクトを作成	ACTIVITYエレメントを作成
作業の順序	TRANSITION	Link	Linkオブジェクトを作成。	TRANSITIONエレメントを作成
作業者	PARTICIPANT HUMAN	User Pool	同名のUserとPoolを作成し、UserをPoolに割り当てる	Userがひとつだけ割り当てられているPoolをHUMANエレメントとして作成。
部門	PARTICIPANT ORGANIZATIONAL UNIT	Pool	部門を表すPoolを作成し、部門に所属するUserを割り当てる	複数Userが所属するPoolをORGANIZATIONAL UNITエレメントとして作成する。
システム	PARTICIPANT SYSTEM	Trigger	Taskの開始に伴ってアプリケーションを起動するTriggerオブジェクトを作成	Triggerの対象アプリケーションをSYSTEMエレメントとして作成
役割	PARTICIPANT ROLE	Role	Roleオブジェクトを作成。	ROLEエレメント作成。
文書	DATA	Document	Documentオブジェクト作成	DATAエレメント作成。
条件	CONDITION	Perform Condition	WPDLクラスのWPDL_CONDITION属性の値として登録	PERFORM_CONDITIONという拡張属性の値として登録。
サブフロー	ACTIVITYが参照するWORKFLOW	Task(階層構造)	ACTIVITYエレメントに対応するTaskのサブTaskとして登録	別のWORKFLOWエレメントとして作成し、Taskに対応するACTIVITYエレメントから参照

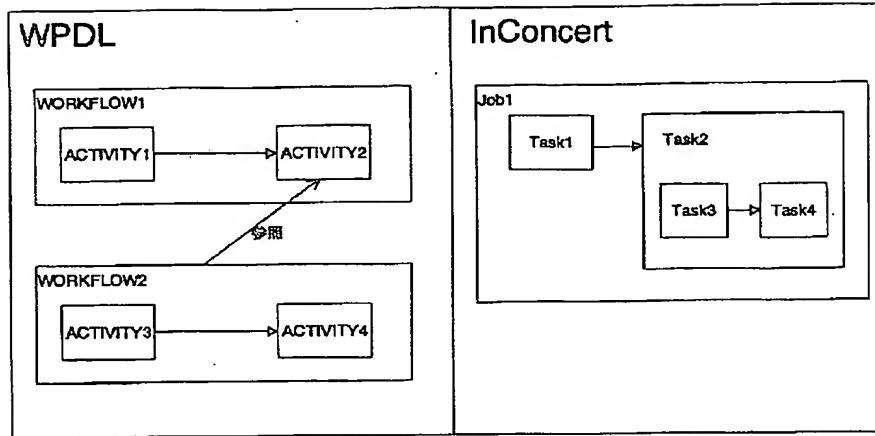
【図13】



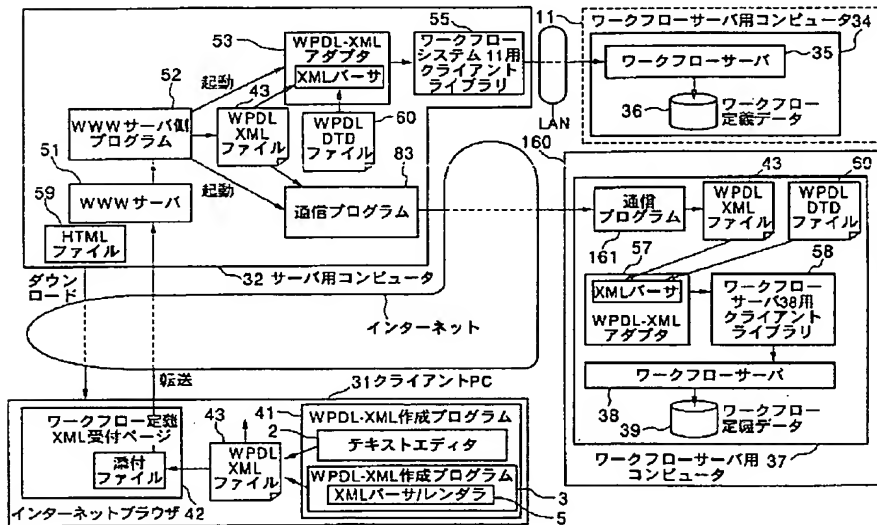
【図14】



【図15】



【図17】



フロントページの続き

(72)発明者 吉本 武弘
東京都港区芝浦1丁目1番1号 東芝ビル
ディング 東芝システム開発株式会社内

Fターム(参考) 5B049 CC21 FF01 GG02
5E501 AC13 BA20 DA02 FA04 FA41